Newton-Krylov-FAC Methods: Infrastructure, Algorithms, and Applications

Michael Pernice[1]
Computer and Computational Sciences Division
Los Alamos National Laboratory
Los Alamos, NM 87544
pernice@lanl.gov

Richard Hornung[2]
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551
hornung@llnl.gov

Seventh Copper Mountain Conference on Iterative Methods
Copper Mountain, Colorado
March 24–29, 2002

[1]This work was performed under the auspices of the U.S. Department of Energy by Los Alamos National Laboratory under contract W-7405-ENG-36. Los Alamos National Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness. LAUR 02-0336

[2]This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-ENG-48.

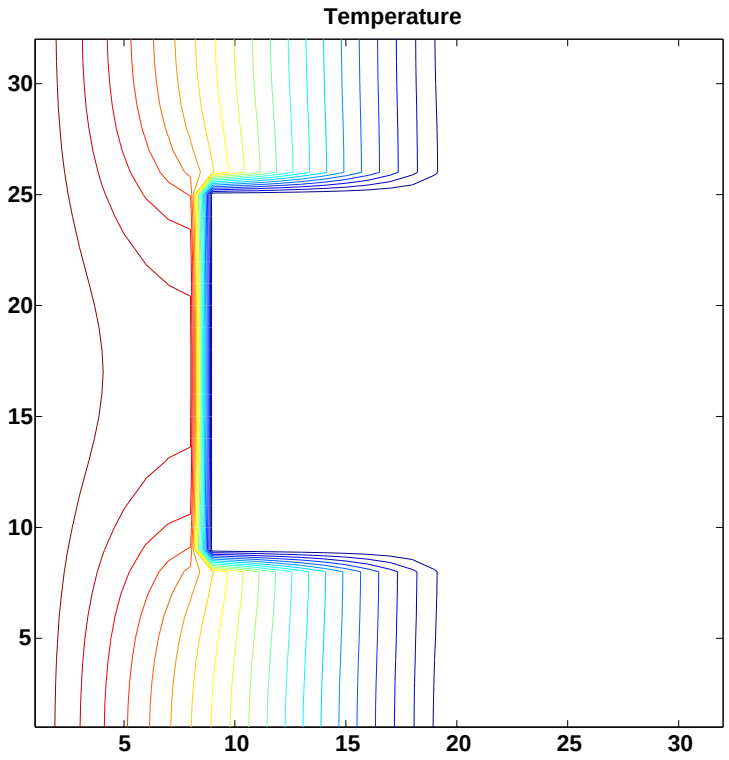
Outline

- Motivation
- Structured adaptive mesh refinement
- Inexact Newton methods
- Software infrastructure
- FAC preconditioning
- Numerical examples

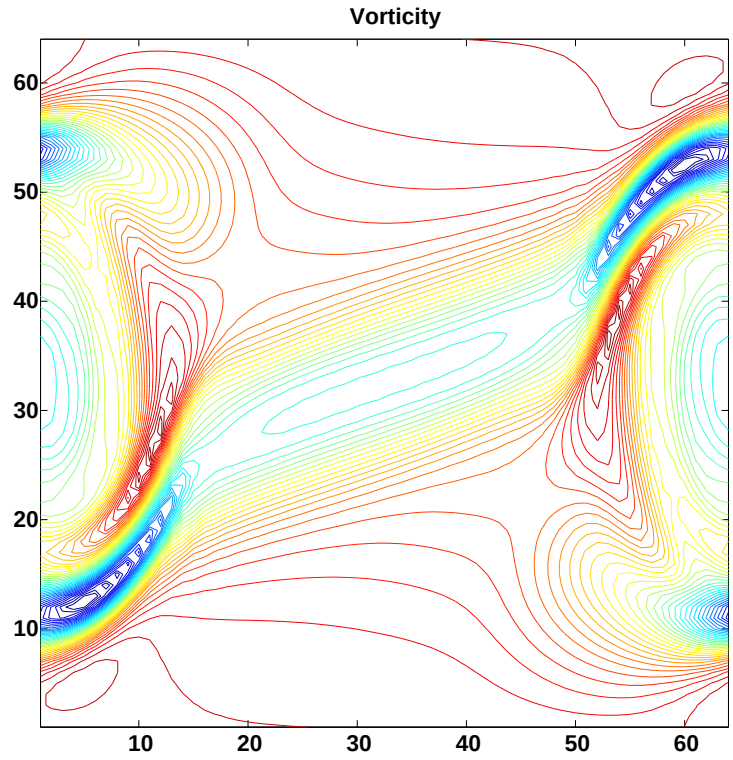
Motivation I: Why Implicit Methods?

- Operator-splitting and/or time-lagging are often used for time-dependent nonlinear problems.
  - Operator-splitting can be used to eliminate nonlinearities, relieve stability constraints, or reduce problem dimensions.
  - Time-lagging techniques use previous time values for some variables to generate linear(ized) problems for other variables.
- Accuracy can be compromised.
  - Nonlinearities not fully converged.
  - Different variables at different time levels introduces additional temporal errors.
- Prior work has shown that *fully implicit* approaches are not only practical, but can *improve accuracy* at a *lower cost*.
  - Time steps are determined solely by accuracy, not stability.
  - Efficient solution of large-scale systems of nonlinear equations is the key.

Motivation II: Why Local Mesh Refinement?



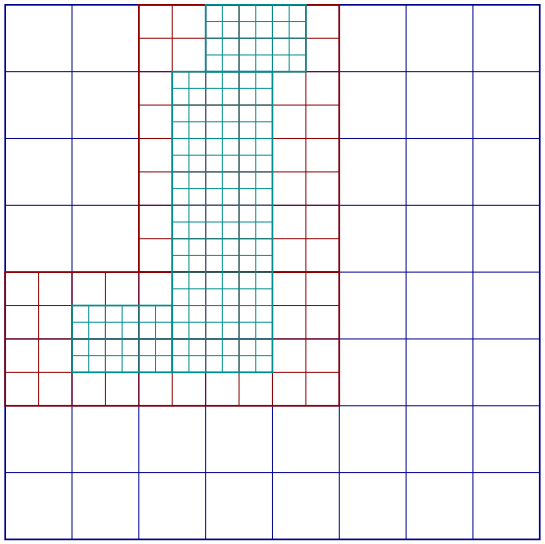
Nonlinear heat transfer



Magnetohydrodynamics

Structured Adaptive Mesh Refinement

Structured adaptive mesh refinement (SAMR) represents a locally refined mesh as a union of logically rectangular meshes.



- The mesh is organized as a hierarchy of nested *refinement levels*.
- Each refinement level defines a region of uniform resolution.
- Each refinement level is the union of logically rectangular *patches*.

Features of SAMR Calculations

- Problem formulation modified only at coarse/fine interfaces.
- Calculations organized primarily as operations on regular uniform grids.
- Inter-/intralevel operations transfer information between refinement levels and supply boundary values for individual patches.
- Grids are generated dynamically by tagging cells on each level that need refinement and grouping these cells into rectangular regions.

Issues to Address

- Handling discretization at coarse/fine interfaces.
- Solver engines.
  - Packages generally are not designed to handle dynamic locally refined grids.
  - Data structures and methods are not "SAMR-aware".
- Preconditioning.
  - Increased resolution should not come at the cost of slower convergence rates.
  - Hierarchical approach facilitates development of preconditioners from simpler building blocks.
- Generality, flexibility, and extensibility.
  - "Best" methods for many problems still a topic for research.
  - Transparent use of different nonlinear solution methods and/or packages.
  - Accommodate a broad range of application requirements.

Jacobian-free Newton-Krylov Methods

The k^{th} step of classical Newton's method requires solution of the Newton equations:

$$F'(x_k)s_k = -F(x_k).$$

With *inexact Newton methods*, we only *approximately* solve the Newton equations

$$\|F(x_k) + F'(x_k)s_k\| \leq \eta_k \|F(x_k)\|.$$

Krylov subspace methods provide a convenient way to determine s_k, since they only need matrix-vector products. The required Jacobian-vector products can be computed using finite differences:

$$F'(x_k)v \approx \frac{F(x_k + \varepsilon v) - F(x_k)}{\varepsilon}.$$

Thus, there is no need to compute and store $F'(x_k)$.

JFNK on SAMR grids

- Core JFNK implementation can be expressed in terms of vector kernel operations.
  - Ideally, these are implemented directly on data represented on the SAMR mesh.
- Nonlinear residual calculation requires detailed knowledge of problem formulation on the SAMR mesh.
  - Ideally, these are implemented in terms of variables from the application.
- Preconditioning is most effective when it is tailored to the problem.
  - Again, direct use of variables from the application is most desirable.
  - Ideally, approaches that work well without local refinement should continue to work well with local refinement
  - *Hierarchical* methods can achieve convergence rates that are independent of the number of refinement levels.

SAMRAI

SAMRAI is a research and development effort in LLNL/CASC that supports the use of SAMR methods in multiphysics problems. It provides a flexible algorithmic framework to explore new solution methods.

Features relevant to our concerns include:

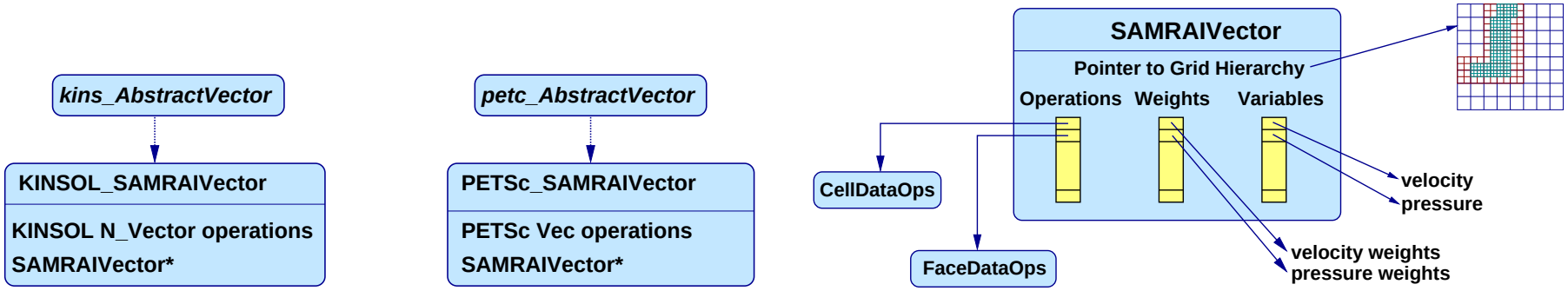
- *Variable management*
  - Supports development in terms of problem variables, independent of grid hierarchy.
- *Vector facilities*
  - Provides a mechanism to group variables together and interact with solver packages.
- *Data transfer facilities*
  - General and flexible means of moving data within the hierarchy.
  - Either built-in interpolation operations or customized schemes can be used.
- *Mesh generation*
  - Supports all aspects of dynamic regridding, including updating vector structures after regridding.

Newton-Krylov Packages

- KINSOL
  - Based on NKSOL.
  - Implemented by Taylor and Hindmarsh at LLNL/CASC.
- SNES
  - Part of the PETSc suite of solvers.
  - Written by Balay, Curfman-McInness, Gropp and Smith at ANL/MCS.
- Many shared features:
  - Jacobian-free methods;
  - globalization based on linesearch backtracking;
  - both static and dynamic choices for forcing terms;
  - flexibility in choosing/updating the preconditioner;
  - modular code organization in which the solver *methods* act on *vector datatypes*.

Interoperability

The design of KINSOL and SNES allows us to use an alternative vector representation and data structure, as long as the required operations (`dot`, `norm`, `axpy`, ...) are available.



With these interfaces, KINSOL or SNES can solve problems defined on SAMR grids.

Interfaces to Solvers and Implicit Timestepping

- Implicit time advancement manages timestepping. Applications provide:
  - description of solution vector;
  - initial conditions;
  - time step computation;
  - time step acceptance/rejection.

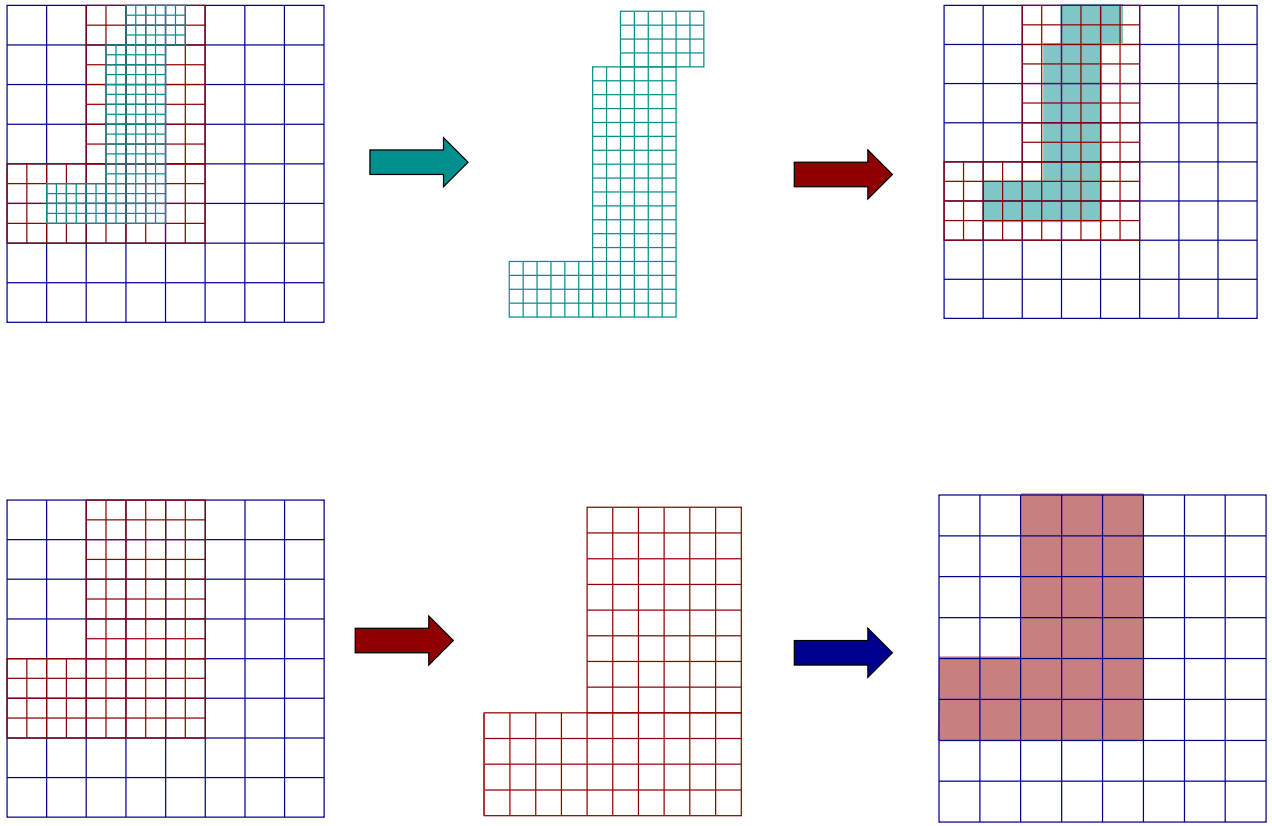
- A uniform interface that allows KINSOL and SNES to be used interchangeably is provided. Applications provide:
  - physical boundary conditions;
  - nonlinear residual evaluation;
  - methods to update Jacobian and compute Jacobian-vector products;
  - methods to set up and apply a preconditioner;

Fast Adaptive Composite Grid Method

Procedure FAC($\underline{h}$, $f^{\underline{h}}$, $u^{\underline{h}}$):

If $\underline{h} = \{h_c\}$, solve $L^{h_c} u^{h_c} = f^{h_c}$ and return.
Set $f^h = I_{\underline{h}}^{h}(f^{\underline{h}} - L^{\underline{h}} u^{\underline{h}})$.
Solve $L^h u^h = f^h$.
Correct $u^{\underline{h}} = u^{\underline{h}} + I_{h}^{\underline{h}} u^h$.
Set $f^{2h} = I_{\underline{h}}^{\underline{2h}}(f^{\underline{h}} - L^{\underline{h}} u^{\underline{h}})$.
$u^{\underline{2h}} = \mathsf{FAC}(\underline{2h}, f^{\underline{2h}}, u^{\underline{2h}})$.
Correct $u^{\underline{h}} = u^{\underline{h}} + I_{\underline{2h}}^{\underline{h}} u^{\underline{2h}}$.
Set $f^h = I_{\underline{h}}^{h}(f^{\underline{h}} - L^{\underline{h}} u^{\underline{h}})$.
Solve $L^h u^h = f^h$.
Correct $u^{\underline{h}} = u^{\underline{h}} + I_{h}^{\underline{h}} u^h$.

FAC Graphical Schematic



FAC Preconditioner Interface

Wide variation in application requirements makes development of a widely applicable FAC preconditioner unlikely.

Observe:

1. the cycling strategy and associated data structures are independent of the application;
2. choice of the preconditioning operator and related methods (smoothing, level solves) are application-specific.

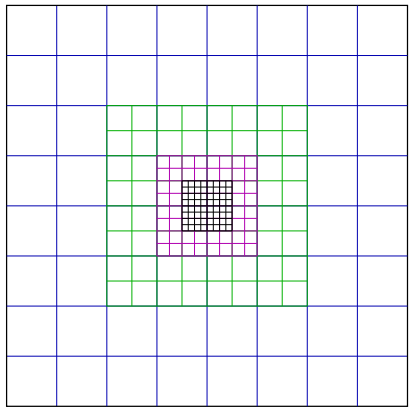
Our approach reflects this division of labor:

- SAMRAI provides
  - an implementation of FAC that separates scheduling of operations from their implementation;
  - additional components that satisfy commonly-encountered needs
    - generic interpolation methods;
    - level solvers based on `hypre` structured solvers.
- Applications provide all problem-specific aspects of the preconditioner:
  - choice of preconditioning operator;
  - smoothing on a level;
  - solving on a level;
  - interlevel transfers, including operator-dependent strategies.

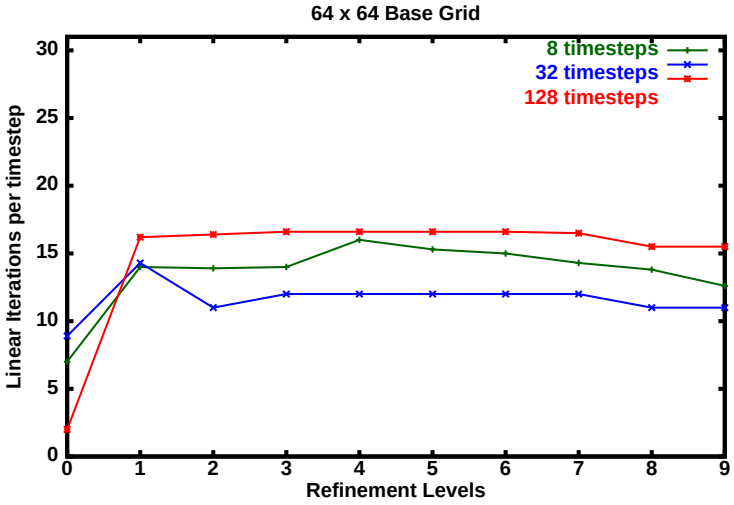
Example: Unsteady Bratu Problem

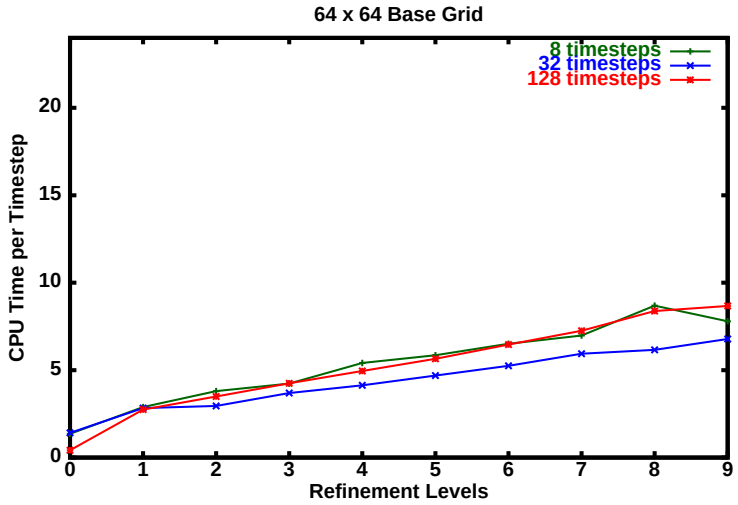
$$u_t = \Delta u + \lambda e^u \text{ on } \Omega = [0,1]^2$$

$$u = 0 \text{ at } t = 0 \text{ and on } \partial\Omega$$



Locally Refined Grid





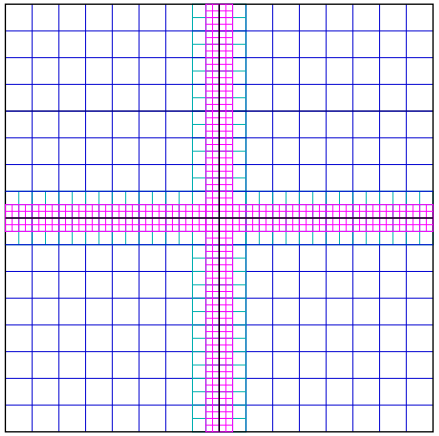
Los Alamos
NATIONAL LABORATORY

Example: Discontinuous Diffusion Coefficients

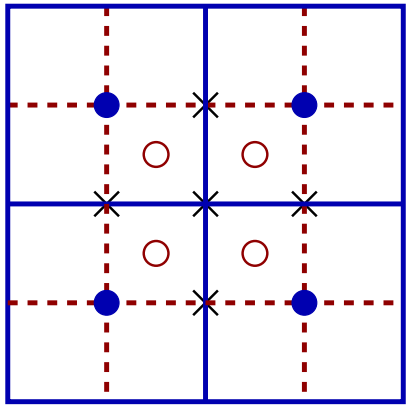
$$u_t = \nabla \cdot D \nabla u + \lambda e^u \text{ on } \Omega = [0,1]^2$$

$$u = 0 \text{ at } t = 0 \text{ and on } \partial\Omega$$

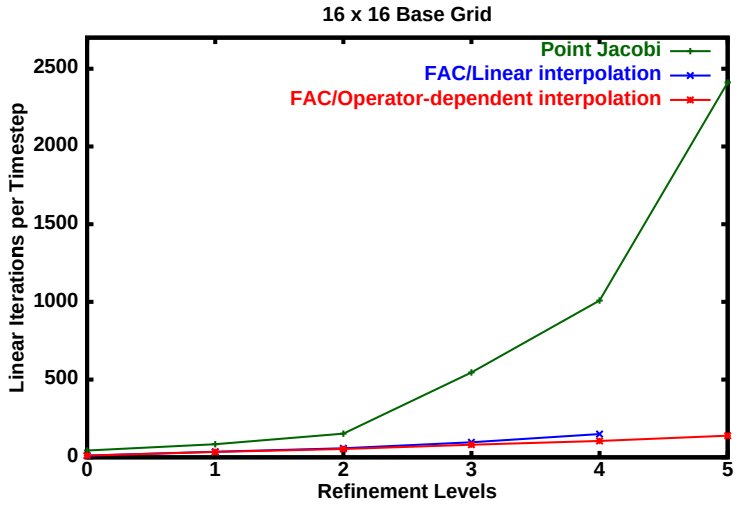
$$D = \begin{cases} 10^{-6} & \text{in } [0,\frac{1}{2}]^2 \cup [\frac{1}{2},1]^2 \\ 1 & \text{otherwise} \end{cases}$$



Locally Refined Grid

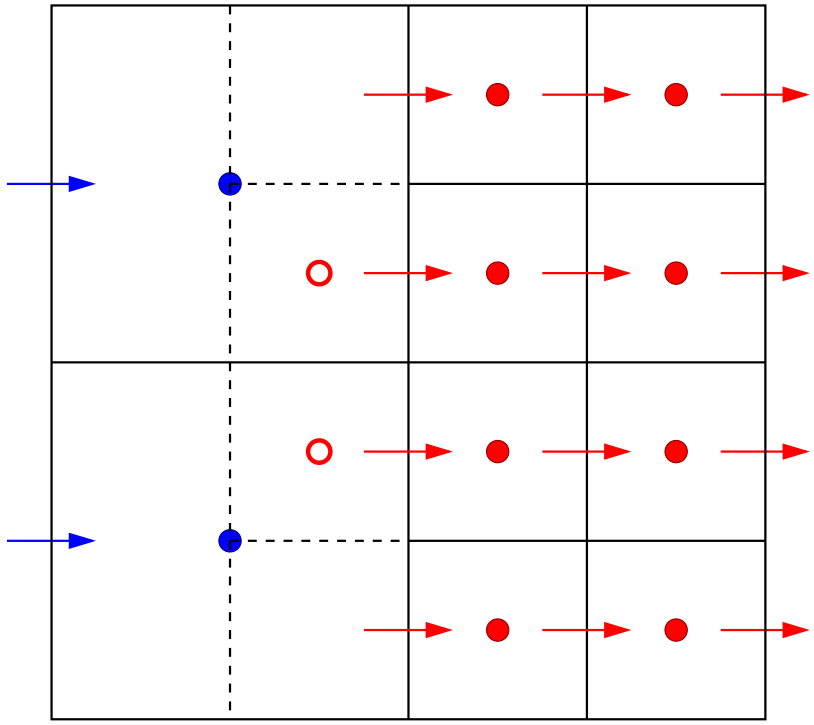


New Interpolation Scheme



Los Alamos NATIONAL LABORATORY

Discretization



Outline of Nonlinear Residual Calculation

```
for each level in the hierarchy (finest to coarsest) {
  for each patch in the level {
    compute face-centered D;
    compute face-centered fluxes D∇u;
  adjust fluxes at coarse/fine interfaces;
  adjust fluxes at physical boundaries;
  }
}
for each level in the hierarchy (finest to coarsest) {
  if not on the finest level {
    copy saved fluxes from finer level to cell faces
      at coarse/fine interfaces;
  }
  for each patch in the level {
    complete evaluation of nonlinear residual,
      including differencing of face fluxes;
  }
}
```


Summary and Future Work

- Developed general and flexible infrastructure that can handle a variety of requirements:
  - different nonlinear solver packages;
  - operator-dependent prolongation;
  - easily changed preconditioners.
- Demonstrated convergence rates independent of number of levels.
- Demonstrate software interoperability among SAMRAI, KINSOL, *hypre* and PETSc.
  - Extensive software reuse allows greater focus on preconditioning.
- Current efforts focused on more realistic applications.
  - Radiation diffusion.
  - Resistive magnetohydrodynamics.
- Preconditioner requirements from applications spur current efforts to "populate the shelf":
  - ✓ simple Jacobi iteration for convection-diffusion operators;
  - ✓ level solver and FAC solver for Poisson problems with Robin boundary conditions;
  - ✓ level solver and FAC solver for convection-diffusion operators;
  - level solver and FAC solver to handle tensor-valued diffusivity.
- Future work:
  - focus on higher order implicit timestepping strategies;
  - balancing temporal and spatial errors.